FaS 14차 세미나

애플리케이션 분석 실습 -Steganography

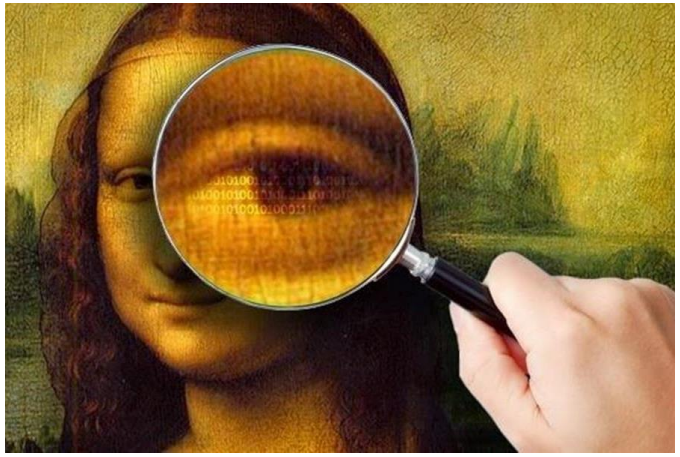
20192233 박진철

CONTENTS

1. Steganography 소개

1. Steganography 소개

스테가노그래피



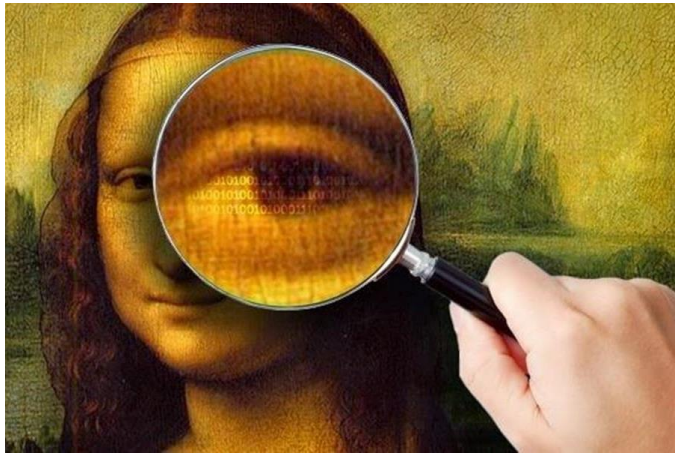
👉 보이는 곳에 비밀을 숨기는 은닉법

👉 디지털 스테가노그래피:
무해해보이는 객체 안에
메시지를 감춰두는 방식

- 보안 프로그램이 이미지나 문서는 통과시킴
→ 통과되는 파일 속에 정보를 은닉

1. Steganography 소개

스테가노그래피



👉 이미지 파일에 은닉

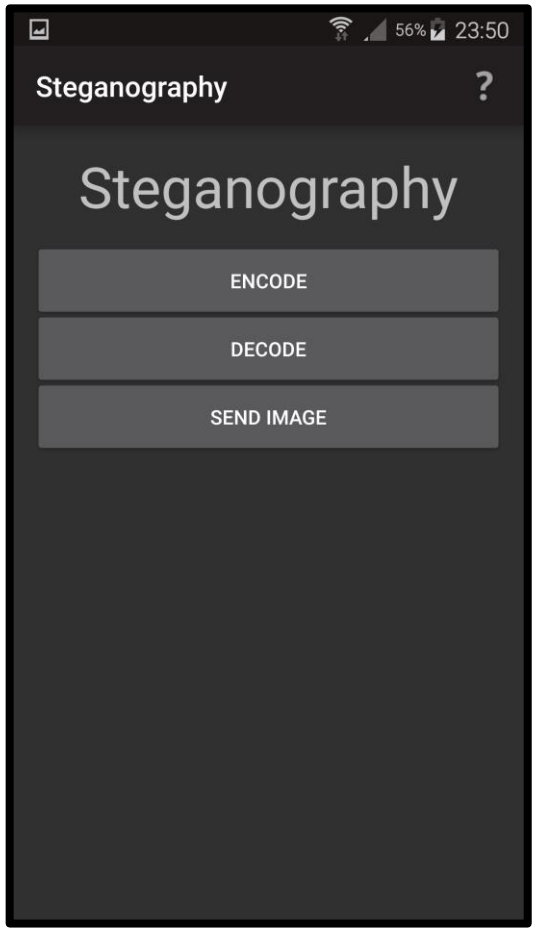
- 각종 도구나 명령어를 통해 데이터 은닉
- 자주 공격에 사용됨
- 세미나에서 분석하는 것도 이미지 파일 방식

👉 문서 파일에 은닉

- 문서 속성정보, 파일 형식에 데이터 은닉
- 은닉된 문서파일을 Office 프로그램으로 실행할 경우, 쉽게 확인할 수 없음

2. Steganography 앱 소개

1. Steganography 앱 소개

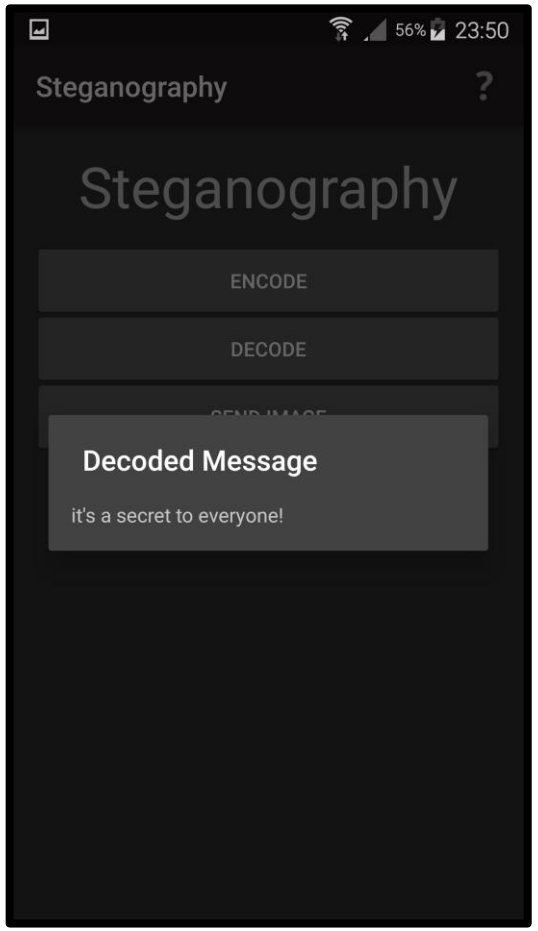


👉 Aksel Tórgarð에서 2016년 개발

👉 이미지에 특정 메시지를 넣어주는 앱

👉메시지가 삽입되어 있는 이미지 복호화

1. Steganography 앱 소개



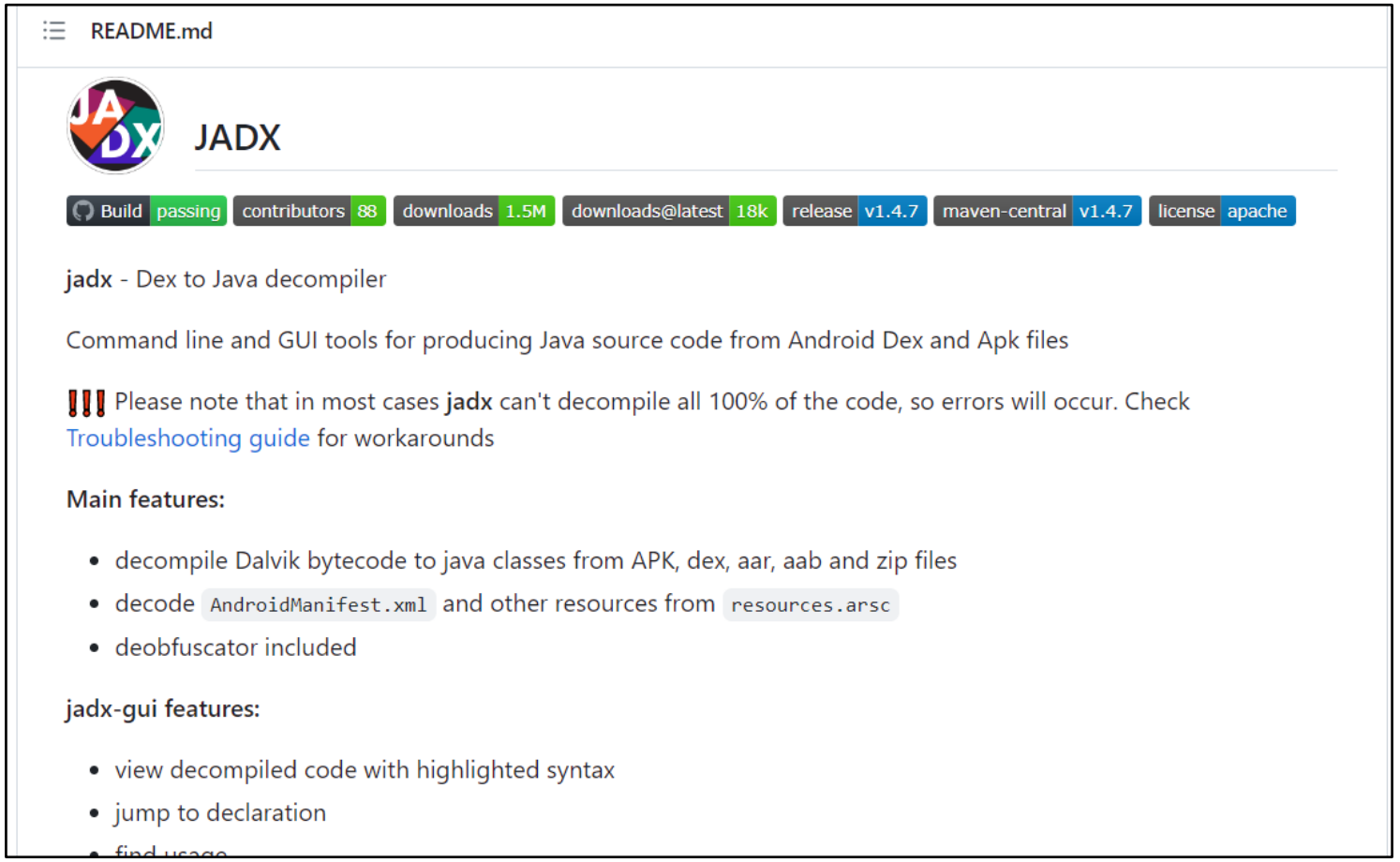
👉 Aksel Tórgarð에서 2016년 개발

👉 이미지에 특정 메시지를 넣어주는 앱

👉메시지가 삽입되어 있는 이미지 복호화

3. Steganography 앱 분석

3. Steganography 앱 분석



README.md

JADX

Build passing | contributors 88 | downloads 1.5M | downloads@latest 18k | release v1.4.7 | maven-central v1.4.7 | license apache

jadx - Dex to Java decompiler

Command line and GUI tools for producing Java source code from Android Dex and Apk files

❗❗❗ Please note that in most cases **jadx** can't decompile all 100% of the code, so errors will occur. Check Troubleshooting guide for workarounds

Main features:

- decompile Dalvik bytecode to java classes from APK, dex, aar, aab and zip files
- decode `AndroidManifest.xml` and other resources from `resources.arsc`
- deobfuscator included

jadx-gui features:

- view decompiled code with highlighted syntax
- jump to declaration
- find usage

👉 jadx를 통해 APK파일 분석

3. Steganography 앱 분석

```java
public static Uri saveBitmap(Bitmap bitmap) {
    File fileFolder = getFileFolder();
    String filename = System.currentTimeMillis() + ".png";
    File file = new File(fileFolder, filename);
    try {
        FileOutputStream out = new FileOutputStream(file);
        bitmap.compress(Bitmap.CompressFormat.PNG, 100, out);
        out.flush();
        out.close();
        return Uri.fromFile(file);
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
```

```java
private static File getFileFolder() {
    String root = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES).toString();
    File fileFolder = new File(root + "/encoded_images");
    fileFolder.mkdirs();
    return fileFolder;
}
```

👉 .png 가 있는 메소드로 이동

3. Steganography 앱 분석

```java
@Override // com.akseltorgard.steganography.async.SteganographyTask
protected SteganographyParams execute(SteganographyParams steganographyParams) {
    Bitmap bitmap = BitmapUtils.decodeFile(steganographyParams.getFilePath());
    int w = bitmap.getWidth();
    int h = bitmap.getHeight();
    int numberOfPixels = w * h;
    byte[] data = steganographyParams.getMessage().getBytes();
    int requiredLength = (data.length * 8) + 32;
    if (requiredLength > numberOfPixels) {
        throw new IllegalArgumentException("Message is too long to fit into pixels.");
    }
    int[] encodedPixels = SteganographyUtils.encode(BitmapUtils.getPixels(bitmap, requiredLength), steganographyParams.getMessage());
    BitmapUtils.setPixels(bitmap, encodedPixels);
    Uri resultUri = FileUtils.saveBitmap(bitmap);
    steganographyParams.setResultUri(resultUri);
    steganographyParams.setType(AsyncResponse.Type.ENCODE_SUCCESS);
    return steganographyParams;
}
```

👉 saveBitmap을 사용하는 메소드로 이동

3. Steganography 앱 분석

```java
@Override // com.akseltorgard.steganography.async.SteganographyTask
protected SteganographyParams execute(SteganographyParams steganographyParams) {
    Bitmap bitmap = BitmapUtils.decodeFile(steganographyParams.getFilePath());
    int w = bitmap.getWidth();
    int h = bitmap.getHeight();
    int numberOfPixels = w * h;
    byte[] data = steganographyParams.getMessage().getBytes();
    int requiredLength = (data.length * 8) + 32;
    if (requiredLength > numberOfPixels) {
        throw new IllegalArgumentException("Message is too long to fit into pixels.");
    }
    int[] encodedPixels = SteganographyUtils.encode(BitmapUtils.getPixels(bitmap, requiredLength), steganographyParams.getMessage());
    BitmapUtils.setPixels(bitmap, encodedPixels);
    Uri resultUri = FileUtils.saveBitmap(bitmap);
    steganographyParams.setResultUri(resultUri);
    steganographyParams.setType(AsyncResponse.Type.ENCODE_SUCCESS);
    return steganographyParams;
}
```

👉 받은 이미지, 메시지의 길이를 확인

3. Steganography 앱 분석

```
@Override // com.akseltorgard.steganography.async.SteganographyTask
protected SteganographyParams execute(SteganographyParams steganographyParams) {
    Bitmap bitmap = BitmapUtils.decodeFile(steganographyParams.getFilePath());
    int w = bitmap.getWidth();
    int h = bitmap.getHeight();
    int numberOfPixels = w * h;
    byte[] data = steganographyParams.getMessage().getBytes();
    int requiredLength = (data.length * 8) + 32;
    if (requiredLength > numberOfPixels) {
        throw new IllegalArgumentException("Message is too long to fit into pixels.");
    }
    int[] encodedPixels = SteganographyUtils.encode(BitmapUtils.getPixels(bitmap, requiredLength), steganographyParams.getMessage());
    BitmapUtils.setPixels(bitmap, encodedPixels);
    Uri resultUri = FileUtils.saveBitmap(bitmap);
    steganographyParams.setResultUri(resultUri);
    steganographyParams.setType(AsyncResponse.Type.ENCODE_SUCCESS);
    return steganographyParams;
}
```

👉 이미지에 메시지를 encode하는 메소드

3. Steganography 앱 분석

```java
@Override // com.akseltorgard.steganography.async.SteganographyTask
protected SteganographyParams execute(SteganographyParams steganographyParams) {
    Bitmap bitmap = BitmapUtils.decodeFile(steganographyParams.getFilePath());
    int w = bitmap.getWidth();
    int h = bitmap.getHeight();
    int numberOfPixels = w * h;
    byte[] data = steganographyParams.getMessage().getBytes();
    int requiredLength = (data.length * 8) + 32;
    if (requiredLength > numberOfPixels) {
        throw new IllegalArgumentException("Message is too long to fit into pixels.");
    }
    int[] encodedPixels = SteganographyUtils.encode(BitmapUtils.getPixels(bitmap, requiredLength), steganographyParams.getMessage());
    BitmapUtils.setPixels(bitmap, encodedPixels);
    Uri resultUri = FileUtils.saveBitmap(bitmap);
    steganographyParams.setResultUri(resultUri);
    steganographyParams.setType(AsyncResponse.Type.ENCODE_SUCCESS);
    return steganographyParams;
}
```

```java
public static int[] encode(int[] pixels, String message) {
    Log.d("Steganography.Encode", "Encode Begin");
    byte[] data = message.getBytes();
    byte[] dataWithLength = new byte[data.length + 4];
    int dataLength = data.length;
    for (int i = 0; i < 4; i++) {
        dataWithLength[i] = (byte) (dataLength & 255);
        dataLength >>>= 8;
    }
    System.arraycopy(data, 0, dataWithLength, 4, data.length);
    int pixelIndex = 0;
    int length = dataWithLength.length;
    for (int i2 = 0; i2 < length; i2++) {
        byte b = dataWithLength[i2];
        int i3 = 0;
        while (i3 < 8) {
            pixels[pixelIndex] = pixels[pixelIndex] & (-2);
            pixels[pixelIndex] = pixels[pixelIndex] | (b & 1);
            b = (byte) (b >>> 1);
            i3++;
            pixelIndex++;
        }
    }
    Log.d("Steganography.Encode", "Encode End");
    return pixels;
}
```

👉 encode() 메소드로 이동

3. Steganography 앱 분석

```
public static int[] encode(int[] pixels, String message) {
    Log.d("Steganography.Encode", "Encode Begin");
    byte[] data = message.getBytes();
    byte[] dataWithLength = new byte[data.length + 4];
    int dataLength = data.length;
    for (int i = 0; i < 4; i++) {
        dataWithLength[i] = (byte) (dataLength & 255);
        dataLength >>>= 8;
    }
    System.arraycopy(data, 0, dataWithLength, 4, data.length);
    int pixelIndex = 0;
    int length = dataWithLength.length;
    for (int i2 = 0; i2 < length; i2++) {
        byte b = dataWithLength[i2];
        int i3 = 0;
        while (i3 < 8) {
            pixels[pixelIndex] = pixels[pixelIndex] & (-2);
            pixels[pixelIndex] = pixels[pixelIndex] | (b & 1);
            b = (byte) (b >>> 1);
            i3++;
            pixelIndex++;
        }
    }
    Log.d("Steganography.Encode", "Encode End");
    return pixels;
}
```

👉 encode() 메소드

3. Steganography 앱 분석

```java
public static int[] encode(int[] pixels, String message) {
    Log.d("Steganography.Encode", "Encode Begin");
    byte[] data = message.getBytes();
    byte[] dataWithLength = new byte[data.length + 4];
    int dataLength = data.length;
    for (int i = 0; i < 4; i++) {
        dataWithLength[i] = (byte) (dataLength & 255);
        dataLength >>>= 8;
    }
    System.arraycopy(data, 0, dataWithLength, 4, data.length);
    int pixelIndex = 0;
    int length = dataWithLength.length;
    for (int i2 = 0; i2 < length; i2++) {
        byte b = dataWithLength[i2];
        int i3 = 0;
        while (i3 < 8) {
            pixels[pixelIndex] = pixels[pixelIndex] & (-2);
            pixels[pixelIndex] = pixels[pixelIndex] | (b & 1);
            b = (byte) (b >>> 1);
            i3++;
            pixelIndex++;
        }
    }
    Log.d("Steganography.Encode", "Encode End");
    return pixels;
}
```

- pixels: 이미지의 픽셀들
- message: 숨기려하는 메시지

encode() 메소드

3. Steganography 앱 분석

```
public static int[] encode(int[] pixels, String message) {
    Log.d("Steganography.Encode", "Encode Begin");
    byte[] data = message.getBytes();
    byte[] dataWithLength = new byte[data.length + 4];
    int dataLength = data.length;
    for (int i = 0; i < 4; i++) {
        dataWithLength[i] = (byte) (dataLength & 255);
        dataLength >>>= 8;
    }
    System.arraycopy(data, 0, dataWithLength, 4, data.length);
    int pixelIndex = 0;
    int length = dataWithLength.length;
    for (int i2 = 0; i2 < length; i2++) {
        byte b = dataWithLength[i2];
        int i3 = 0;
        while (i3 < 8) {
            pixels[pixelIndex] = pixels[pixelIndex] & (-2);
            pixels[pixelIndex] = pixels[pixelIndex] | (b & 1);
            b = (byte) (b >>> 1);
            i3++;
            pixelIndex++;
        }
    }
    Log.d("Steganography.Encode", "Encode End");
    return pixels;
}
```

👉 메시지를 바이트 단위로 가져옴
👉 가져온 값을 data에 저장

👉 encode() 메소드

3. Steganography 앱 분석

```
public static int[] encode(int[] pixels, String message) {
    Log.d("Steganography.Encode", "Encode Begin");
    byte[] data = message.getBytes();
    byte[] dataWithLength = new byte[data.length + 4];
    int dataLength = data.length;
    for (int i = 0; i < 4; i++) {
        dataWithLength[i] = (byte) (dataLength & 255);
        dataLength >>>= 8;
    }
    System.arraycopy(data, 0, dataWithLength, 4, data.length);
    int pixelIndex = 0;
    int length = dataWithLength.length;
    for (int i2 = 0; i2 < length; i2++) {
        byte b = dataWithLength[i2];
        int i3 = 0;
        while (i3 < 8) {
            pixels[pixelIndex] = pixels[pixelIndex] & (-2);
            pixels[pixelIndex] = pixels[pixelIndex] | (b & 1);
            b = (byte) (b >>> 1);
            i3++;
            pixelIndex++;
        }
    }
    Log.d("Steganography.Encode", "Encode End");
    return pixels;
}
```

👉 data의 길이 + 4만큼의 길이를 가진 dataWithLength 생성

👉 data의 길이를 dataLength에 저장

👉 encode() 메소드

3. Steganography 앱 분석

```
public static int[] encode(int[] pixels, String message) {
    Log.d("Steganography.Encode", "Encode Begin");
    byte[] data = message.getBytes();
    byte[] dataWithLength = new byte[data.length + 4];
    int dataLength = data.length;
    for (int i = 0; i < 4; i++) {
        dataWithLength[i] = (byte) (dataLength & 255);
        dataLength >>>= 8;
    }
    System.arraycopy(data, 0, dataWithLength, 4, data.length);
    int pixelIndex = 0;
    int length = dataWithLength.length;
    for (int i2 = 0; i2 < length; i2++) {
        byte b = dataWithLength[i2];
        int i3 = 0;
        while (i3 < 8) {
            pixels[pixelIndex] = pixels[pixelIndex] & (-2);
            pixels[pixelIndex] = pixels[pixelIndex] | (b & 1);
            b = (byte) (b >>> 1);
            i3++;
            pixelIndex++;
        }
    }
    Log.d("Steganography.Encode", "Encode End");
    return pixels;
}
```

- 32비트 정수 dataLength를 4개의 8바이트로 나누어 dataWithLength 배열에 저장
- 저장된 바이트는 배열의 처음 4개 원소에 차례대로 들어감

encode() 메소드

3. Steganography 앱 분석

```java
@Override // com.akseltorgard.steganography.async.SteganographyTask
protected SteganographyParams execute(SteganographyParams steganographyParams) {
    Bitmap bitmap = BitmapUtils.decodeFile(steganographyParams.getFilePath());
    int w = bitmap.getWidth();
    int h = bitmap.getHeight();
    int numberOfPixels = w * h;
    byte[] data = steganographyParams.getMessage().getBytes();
    int requiredLength = (data.length * 8) + 32;
    if (requiredLength > numberOfPixels) {
        throw new IllegalArgumentException("Message is too long to fit into pixels.");
    }
    int[] encodedPixels = SteganographyUtils.encode(BitmapUtils.getPixels(bitmap, requiredLength), steganographyParams.getMessage());
    BitmapUtils.setPixels(bitmap, encodedPixels);
    Uri resultUri = FileUtils.saveBitmap(bitmap);
    steganographyParams.setResultUri(resultUri);
    steganographyParams.setType(AsyncResponse.Type.ENCODE_SUCCESS);
    return steganographyParams;
}
```

👉 받은 이미지, 메시지의 길이를 확인

3. Steganography 앱 분석

```
public static int[] encode(int[] pixels, String message) {
    Log.d("Steganography.Encode", "Encode Begin");
    byte[] data = message.getBytes();
    byte[] dataWithLength = new byte[data.length + 4];
    int dataLength = data.length;
    for (int i = 0; i < 4; i++) {
        dataWithLength[i] = (byte) (dataLength & 255);
        dataLength >>>= 8;
    }
    System.arraycopy(data, 0, dataWithLength, 4, data.length);
    int pixelIndex = 0;
    int length = dataWithLength.length;
    for (int i2 = 0; i2 < length; i2++) {
        byte b = dataWithLength[i2];
        int i3 = 0;
        while (i3 < 8) {
            pixels[pixelIndex] = pixels[pixelIndex] & (-2);
            pixels[pixelIndex] = pixels[pixelIndex] | (b & 1);
            b = (byte) (b >>> 1);
            i3++;
            pixelIndex++;
        }
    }
    Log.d("Steganography.Encode", "Encode End");
    return pixels;
}
```

- 32비트 정수 dataLength를 4개의 8바이트로 나누어 dataWithLength 배열에 저장
- 저장된 바이트는 배열의 처음 4개 원소에 차례대로 들어감

encode() 메소드

3. Steganography 앱 분석

```java
public static int[] encode(int[] pixels, String message) {
    Log.d("Steganography.Encode", "Encode Begin");
    byte[] data = message.getBytes();
    byte[] dataWithLength = new byte[data.length + 4];
    int dataLength = data.length;
    for (int i = 0; i < 4; i++) {
        dataWithLength[i] = (byte) (dataLength & 255);
        dataLength >>>= 8;
    }
    System.arraycopy(data, 0, dataWithLength, 4, data.length);
    int pixelIndex = 0;
    int length = dataWithLength.length;
    for (int i2 = 0; i2 < length; i2++) {
        byte b = dataWithLength[i2];
        int i3 = 0;
        while (i3 < 8) {
            pixels[pixelIndex] = pixels[pixelIndex] & (-2);
            pixels[pixelIndex] = pixels[pixelIndex] | (b & 1);
            b = (byte) (b >>> 1);
            i3++;
            pixelIndex++;
        }
    }
    Log.d("Steganography.Encode", "Encode End");
    return pixels;
}
```

👉 arraycopy(src, srcPos, dest, destPos, length)
- src: 복사할 원본 배열
- srcPos: 원본 배열에서 복사를 시작할 인덱스 위치
- dest: 복사된 요소를 저장할 대상 배열
- destPos: 대상 배열에서 복사를 시작할 인덱스 위치
- length: 복사할 요소의 수

👉 data를 dataWithLength에 저장

👉 단, dataWithLength의 처음 4개의 인덱스에는 data의 길이 정보가 저장

⇒ 숨겨진 데이터의 길이를 알 수 있음

👉 encode() 메소드

3. Steganography 앱 분석

```
public static int[] encode(int[] pixels, String message) {
    Log.d("Steganography.Encode", "Encode Begin");
    byte[] data = message.getBytes();
    byte[] dataWithLength = new byte[data.length + 4];
    int dataLength = data.length;
    for (int i = 0; i < 4; i++) {
        dataWithLength[i] = (byte) (dataLength & 255);
        dataLength >>>= 8;
    }
    System.arraycopy(data, 0, dataWithLength, 4, data.length);
    int pixelIndex = 0;
    int length = dataWithLength.length;
    for (int i2 = 0; i2 < length; i2++) {
        byte b = dataWithLength[i2];
        int i3 = 0;
        while (i3 < 8) {
            pixels[pixelIndex] = pixels[pixelIndex] & (-2);
            pixels[pixelIndex] = pixels[pixelIndex] | (b & 1);
            b = (byte) (b >>> 1);
            i3++;
            pixelIndex++;
        }
    }
    Log.d("Steganography.Encode", "Encode End");
    return pixels;
}
```

- pixelIndex: 이미지의 픽셀 인덱스
- length에 dataWithLength의 길이 저장

- encode() 메소드

3. Steganography 앱 분석

```
public static int[] encode(int[] pixels, String message) {
    Log.d("Steganography.Encode", "Encode Begin");
    byte[] data = message.getBytes();
    byte[] dataWithLength = new byte[data.length + 4];
    int dataLength = data.length;
    for (int i = 0; i < 4; i++) {
        dataWithLength[i] = (byte) (dataLength & 255);
        dataLength >>>= 8;
    }
    System.arraycopy(data, 0, dataWithLength, 4, data.length);
    int pixelIndex = 0;
    int length = dataWithLength.length;
    for (int i2 = 0; i2 < length; i2++) {
        byte b = dataWithLength[i2];
        int i3 = 0;
        while (i3 < 8) {
            pixels[pixelIndex] = pixels[pixelIndex] & (-2);
            pixels[pixelIndex] = pixels[pixelIndex] | (b & 1);
            b = (byte) (b >>> 1);
            i3++;
            pixelIndex++;
        }
    }
    Log.d("Steganography.Encode", "Encode End");
    return pixels;
}
```

👉 dataWithLength 길이만큼 반복 수행

👉 b에 dataWithLength[i2] 저장

👉 encode() 메소드

3. Steganography 앱 분석

```
public static int[] encode(int[] pixels, String message) {
    Log.d("Steganography.Encode", "Encode Begin");
    byte[] data = message.getBytes();
    byte[] dataWithLength = new byte[data.length + 4];
    int dataLength = data.length;
    for (int i = 0; i < 4; i++) {
        dataWithLength[i] = (byte) (dataLength & 255);
        dataLength >>>= 8;
    }
    System.arraycopy(data, 0, dataWithLength, 4, data.length);
    int pixelIndex = 0;
    int length = dataWithLength.length;
    for (int i2 = 0; i2 < length; i2++) {
        byte b = dataWithLength[i2];
        int i3 = 0;
        while (i3 < 8) {
            pixels[pixelIndex] = pixels[pixelIndex] & (-2);
            pixels[pixelIndex] = pixels[pixelIndex] | (b & 1);
            b = (byte) (b >>> 1);
            i3++;
            pixelIndex++;
        }
    }
    Log.d("Steganography.Encode", "Encode End");
    return pixels;
}
```

👉 b의 각 비트를 처리하기 위한 루프 시작
(1바이트 = 8비트)

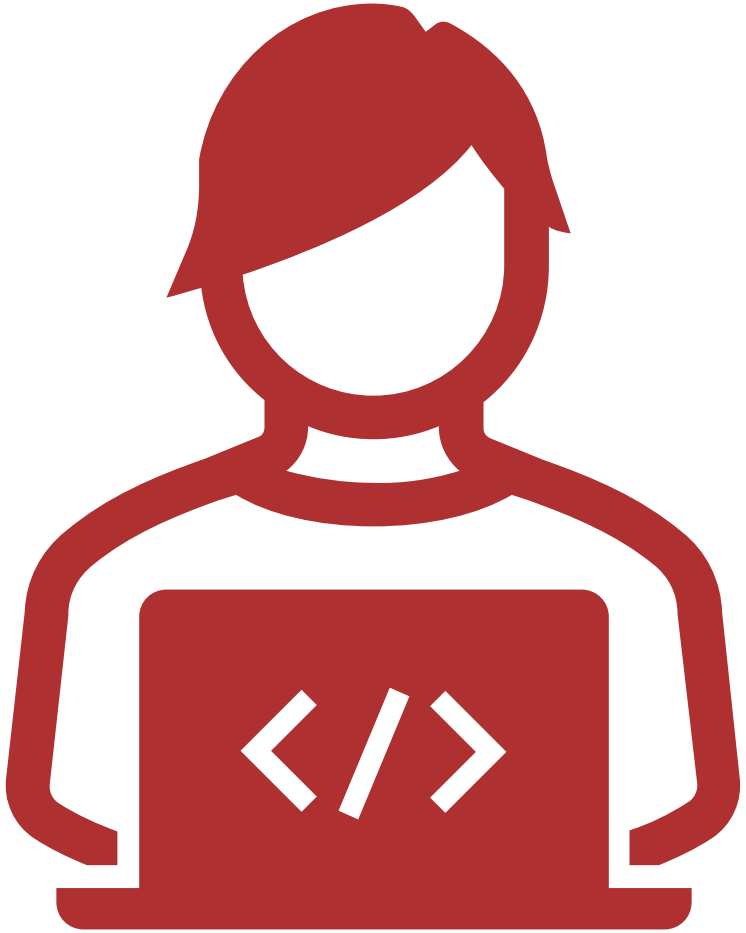
👉 -2는 0b11111110
→ pixels[pixelIndex]의 가장 낮은 비트를 0으로 만듦
→ 픽셀 값을 짝수로 만들어줌

👉 pixels[pixelIndex]의 가장 낮은 비트를 b의 가장 낮은 비트로 설정
- (b&1): b의 가장 낮은 비트를 추출

👉 b를 오른쪽으로 1비트 시프트

⇒ b(바이트)의 각 비트가 이미지의 각 픽셀의 가장 낮은 비트에 숨겨짐

👉 encode() 메소드



4. 스테가노그래피 코드 구현(미완성)

4. 스테가노그래피 코드 구현

```
"""
=============================================================================================
encode: 이미지에 메세지를 숨기는 함수

input
image_path: 이미지의 경로
message: 숨길 메세지
=============================================================================================
"""
def encode(image_path, message):
    #![1]이미지를 열어서 RGB로 변환
    img = Image.open(image_path)
    img = img.convert('RGB')
    pixels = np.array(img)

    #![2]메세지를 바이트로 변환
    data = message.encode('utf-8')

    #![3]메세지의 길이+4 만큼의 배열을 생성
    data_with_length = np.zeros((len(data) + 4), dtype=np.uint8)

    data_length = len(data) #메세지의 길이

    #![4]메세지의 길이를 data_with_length에 저장
    for i in range(4): #32비트 정수 data_length를 8비트 정수 4개로 나누어 저장
        #8바이트씩 저장
        data_with_length[i] = data_length & 255
        data_length >>= 8

    #![5]메세지를 data_with_length에 저장
    for i in range(len(data)):
        data_with_length[i+4] = data[i]

    #![6]이미지의 픽셀 인덱스 초기화 data_with_length의 길이 저장
    pixel_index = 0
    length = len(data_with_length) #메세지의 길이+4

    #![7]data_with_length를 픽셀에 저장
    for i in range(length): #data_with_length의 길이만큼 반복
        #![7-1]data_with_length의 i번째 바이트를 b에 저장
        b=data_with_length[i]

        #![7-2]b의 8비트를 픽셀에 한 비트씩 저장
        for j in range(8):
            #픽셀의 마지막 비트를 0으로 초기화
            pixels[pixel_index] = pixels[pixel_index] & (-2)
            #픽셀의 마지막 비트를 b의 마지막 비트로 저장
            pixels[pixel_index] = pixels[pixel_index] | (b & 1)
            #b를 오른쪽으로 1비트씩 이동
            b >>= 1
            pixel_index += 1

    #![8]이미지를 저장
    img = Image.fromarray(pixels)
    img.save('encoded.png')
```

4. 스테가노그래피 코드 구현

```
"""
================================================================================================
encode: 이미지에 메세지를 숨기는 함수

input
image_path: 이미지의 경로
message: 숨길 메세지
================================================================================================
"""
def encode(image_path, message):
```

4. 스테가노그래피 코드 구현

```
#![1]이미지를 열어서 RGB로 변환
img = Image.open(image_path)
img = img.convert('RGB')
pixels = np.array(img)

#![2]메세지를 바이트로 변환
data = message.encode('utf-8')
```

4. 스테가노그래피 코드 구현

```
#![3]메세지의 길이+4 만큼의 배열을 생성
data_with_length = np.zeros((len(data) + 4), dtype=np.uint8)

data_length = len(data) #메세지의 길이

#![4]메세지의 길이를 data_with_length에 저장
for i in range(4): #32비트 정수 data_length를 8비트 정수 4개로 나누어 저장
    #8바이트씩 저장
    data_with_length[i] = data_length & 255
    data_length >>= 8
```

4. 스테가노그래피 코드 구현

```
#![5]메세지를 data_with_length에 저장
for i in range(len(data)):
    data_with_length[i+4] = data[i]

#![6]이미지의 픽셀 인덱스 초기화 data_with_length의 길이 저장
pixel_index = 0
length = len(data_with_length) #메세지의 길이+4
```

4. 스테가노그래피 코드 구현

```
#![7]data_with_length를 픽셀에 저장
for i in range(length): #data_with_length의 길이만큼 반복
    #![7-1]data_with_length의 i번째 바이트를 b에 저장
    b=data_with_length[i]

    #![7-2]b의 8비트를 픽셀에 한 비트씩 저장
    for j in range(8):
        #픽셀의 마지막 비트를 0으로 초기화
        pixels[pixel_index] = pixels[pixel_index] & (-2)
        #픽셀의 마지막 비트를 b의 마지막 비트로 저장
        pixels[pixel_index] = pixels[pixel_index] | (b & 1)
        #b를 오른쪽으로 1비트씩 이동
        b >>= 1
        pixel_index += 1

#![8]이미지를 저장
img = Image.fromarray(pixels)
img.save('encoded.png')
```

4. 스테가노그래피 코드 구현

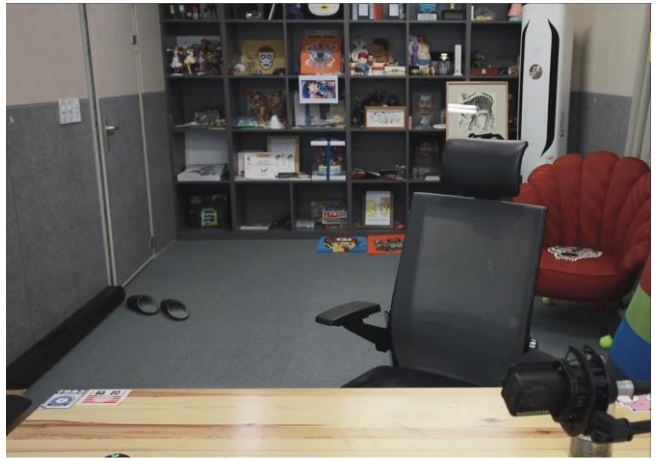
test.png

↓

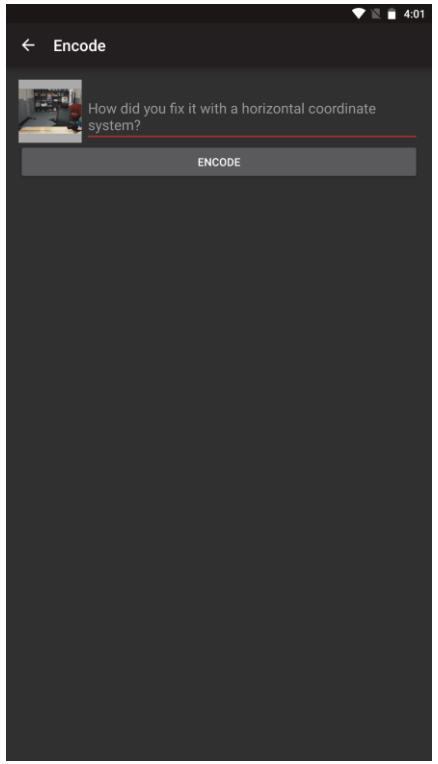
```
def main():
    encode('test.png', 'How did you fix it with a horizontal coordinate system?')

if __name__ == '__main__':
    main()
```

4. 스테가노그래피 코드 구현

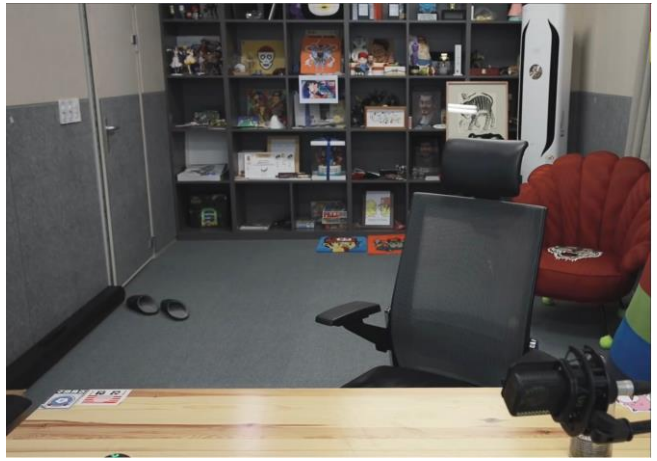
test.png



encoded.png



test.png



1699988515978.png

4. 스테가노그래피 코드 구현

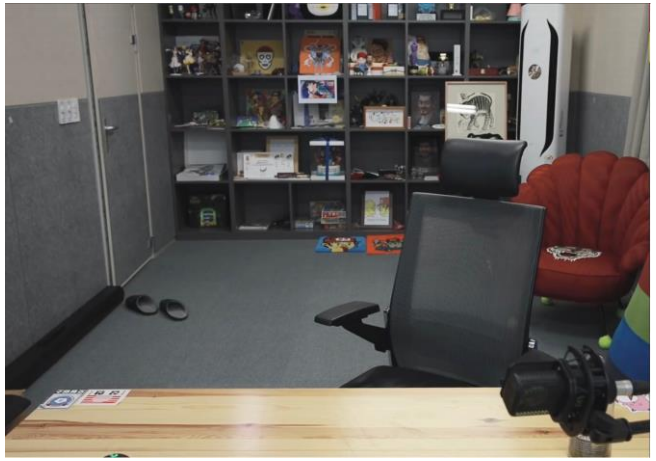
encoded.png

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	‰PNG........IHDR
00000010	00	00	0A	04	00	00	07	0C	08	02	00	00	00	ED	C8	ED	..............íÈí
00000020	4E	00	01	00	00	49	44	41	54	78	9C	EC	FD	DB	8E	24	N....IDATxœìýÛŽ$
00000030	39	B2	20	08	0A	45	E9	1A	9A	56	56	7E	BC	03	89	44	9² ..Eé.šVV~¼.‰D
00000040	E1	60	D0	38	18	9C	87	7E	DA	C7	7D	DA	FF	E8	3F	DB	á`Ð8.œ‡~ÚÇ}Úÿè?Û
00000050	EF	5A	0C	06	83	C6	60	D1	0F	8D	46	21	11	9D	1B	27	ïZ..ƒÆ`Ñ..F!...'
00000060	CA	CB	D3	42	5D	9D	2A	DC	07	B9	50	48	A5	9A	99	7B	ÊËÓB].*Ü.¹PH¥š™{
00000070	78	5C	B2	CA	25	23	23	CC	D4	A8	BC	08	85	72	25	85	x\²Ê%##ÌÔ¨¼.…r%…
00000080	E1	BF	FE	D7	FF	0A	6F	F0	06	6F	F0	06	6F	F0	06	6F	á¿þ×ÿ.oð.oð.oð.o
00000090	F0	06	6F	F0	06	6F	F0	06	6F	F0	06	6F	F0	06	6F	F0	ð.oð.oð.oð.oð.oð
000000A0	06	6F	F0	06	6F	F0	06	6F	F0	06	6F	F0	06	6F	F0	06	.oð.oð.oð.oð.oð.
000000B0	6F	F0	8F	05	F8	BD	3B	F0	06	6F	F0	06	6F	F0	06	6F	oð..ø½;ð.oð.oð.o
000000C0	F0	06	6F	F0	06	6F	F0	06	6F	F0	06	6F	F0	06	6F	F0	ð.oð.oð.oð.oð.oð
000000D0	06	6F	F0	06	6F	F0	06	6F	F0	06	6F	F0	06	6F	F0	06	.oð.oð.oð.oð.oð.
000000E0	6F	F0	06	AF	0F	F1	FF	F5	FF	FC	7F	EC	FE	48	40	40	oð.¯.ñÿõÿü.ìþH@@
000000F0	DD	5F	B0	2A	D5	7F	BE	0F	08	08	40	FA	1E	22	00	00	Ý_°*Õ.¾...@ú."..
00000100	11	81	3E	44	EC	D4	D3	7B	F6	2A	80	56	3F	41	3D	9A	..>DìÔÓ{ö*€V?A=š

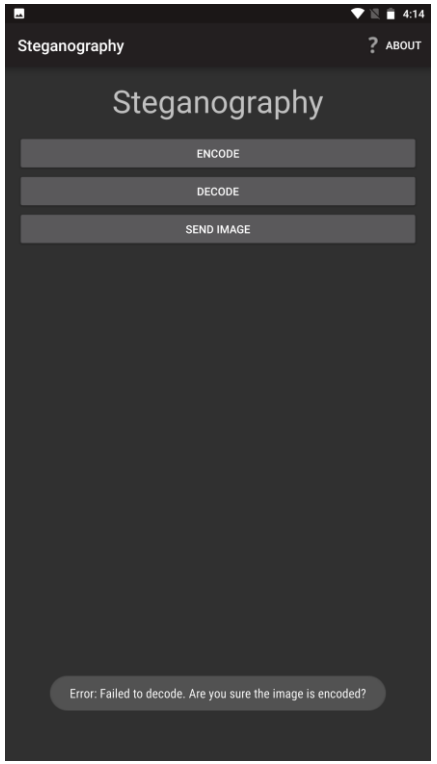
1699988515978.png

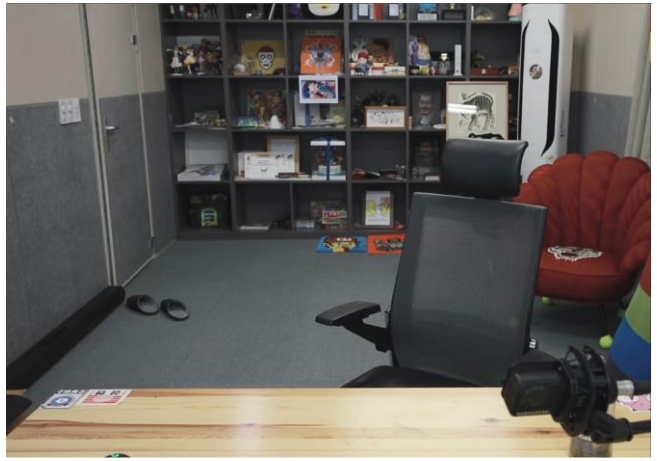
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	‰PNG........IHDR
00000010	00	00	0A	04	00	00	07	0C	08	06	00	00	00	62	AA	7A	.............bªz
00000020	19	00	00	00	04	73	42	49	54	08	08	08	08	7C	08	64	.....sBIT....|.d
00000030	88	00	00	20	00	49	44	41	54	78	9C	EC	BD	C9	92	24	ˆ.. .IDATxœì½É'$
00000040	49	92	25	F6	58	54	D5	16	5F	23	22	97	AA	2E	60	9A	I'%öXTÕ._#"—ª.`š
00000050	BA	07	4D	D4	37	1C	71	C3	6F	0C	FE	0C	FF	D5	07	10	º.MÔ7.qÃo.þ.ÿÕ..
00000060	66	40	D4	38	0C	4D	63	50	93	98	CA	25	C2	DD	CD	4C	f@Ô8.McP"˜Ê%ÂÝÍL
00000070	55	84	71	E0	45	58	D4	D4	7C	89	3D	32	8D	33	3D	CC	U„qàEXÔÔ|‰=2.3=Ì
00000080	4C	57	59	58	78	93	27	2C	F4	1F	FE	C3	FF	C6	70	62	LWYXx"',ô.þÃÿÆpb
00000090	00	14	BE	7F	6E	A2	F0	5E	FB	4E	B3	6B	E6	65	A4	13	..¾.n¢ð^ûN³kæe¤.
000000A0	E7	EC	F7	53	EF	39	F5	EE	F8	BC	53	65	8A	CF	38	75	çì÷Sï9õîø¼SeŠÏ8u
000000B0	FF	53	EF	5F	BA	2F	96	FF	39	F5	9F	7F	C6	6B	9F	7A	ÿSï_º/–ÿ9õŸ.ÆkŸz
000000C0	FF	52	79	DF	A7	FE	A7	CA	FF	D4	FB	5F	DA	BE	2F	AD	ÿRyß§þ§ÊÿÔû_Ú¾/.
000000D0	DF	53	FC	F1	31	DF	FF	54	FB	9C	2A	DF	A9	76	7C	09	ßSüñ1ßÿTûœ*ß©v|.
000000E0	FF	9E	7A	FF	53	FC	B3	54	9E	F9	B5	4F	B5	CF	73	C6	ÿžzÿSü³TžùµOµÏsÆ
000000F0	EA	53	EF	7F	EC	FC	A9	77	CC	CB	17	AF	5D	BA	FF	B1	êSï.ìü©wÌË.¯]ºÿ±
00000100	E7	CF	EF	7F	6C	2C	BF	A4	7D	3F	B4	FD	9F	2A	D3	73	çÏï.l,¿¤}?´ýŸ*Ós

4. 스테가노그래피 코드 구현

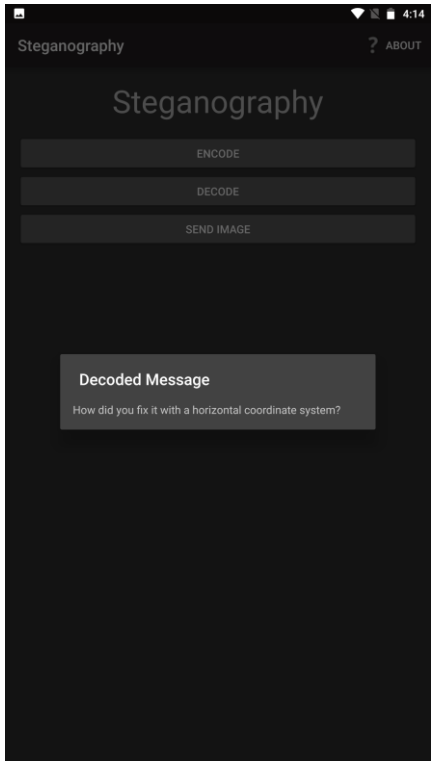


encoded.png





1699988515978.png



THANKS TO WATCHING

20192233 박진철